ESD-TR-72-157

MTR-2303

CURRENT TRENDS IN DATA MANAGEMENT SYSTEM ARCHITECTURE

J. A. Singer

MARCH 1973

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 572M
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract F19 628 -71-C-0002

ESD-TR-72-157

MTR-2303

CURRENT TRENDS IN DATA MANAGEMENT SYSTEM ARCHITECTURE

J. A. Singer

MARCH 1973

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 572M

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract F19 628 -71-C-0002

FOREWORD

This report has been prepared by The MITRE Corporation under Project 572M of Contract F19(628)-71-C-0002. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, L. G. Hanscom Field, Bedford, Massachusetts.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

MELVIN B. EMMONS, Colonel, USAF
Director, Information Systems Division
Deputy for Command & Management Systems

ABSTRACT

Early data management system architecture is examined and compared with that of current systems. This comparison reveals a trend toward placing a number of basic data management system components within the modern operating system. A continuation of this trend is postulated and a number of specific examples of areas where the trend is likely to continue are given. The advantages of a continuation of this trend are described in terms of both the ease of building new data management systems and the increased compatibility of such systems with other data processing tools.

PREFACE

This paper was presented at the USAF Academy Worldwide Data Management Symposium on Design and Evaluation, held at the Air Force Academy on 9-11 December 1971.

TABLE OF CONTENTS

INTRODUCTION

The views expressed in this paper are derived to a great extent from the author's involvement in both past and present data management projects sponsored by the Electronic Systems Division of the Air Force. These projects span a period of more than ten years during which there has been considerable change in the field of data management. The most significant single change is the growing acceptance of data management as an essential set of tools for the management of formatted files. Ten years ago, virtually all significant data management development work was being done within the Department of Defense community, largely in response to the demanding requirements of command and control systems. The management support users of data processing, both in the military and commercial environments, regarded data management systems as being inappropriate tools for their immediate needs. The reasons for this are fairly straightforward. First, the demands for high speed data retrieval and multi-user access were critical needs in the command and control environment and strongly encouraged the development of such systems. Second, the cost of such systems was very high. They typically required over a million dollars to implement, and took a number of years to complete. This level of investment was out of range with that which the management support community could afford. Additional impetus for the building of these large, high-performance systems was provided through the availability in the command and control environment of what was then large sophisticated hardware such as the SAGE computers. Many non-command and control users of data processing did show an interest in data management, however, their available resources limited this interest to using fairly simple tape file extraction and report generation programs.

This early situation is in strong contrast with that of the present. There are over one hundred data management systems being marketed today largely as proprietary software, and there is a dramatic increase in the use of these systems by commercial users of data processing. The Informatics Inc. MARK IV system, for example, has a client list of over 400, while IBM's IMS system is over 200. Within the set of DOD systems, NIPS has found wide acceptance in terms of the number of installations where it is being used.

The most significant conclusion to be drawn from these numbers is that data management is no longer regarded as the exclusive property of command and control, but as a necessary and even vital part of other segments of the data processing user community. More specifically, the management support segment which is dominant in both the military and commercial environments has begun to greatly expand its interest in and use of sophisticated data management capabilities.

This wide use, more than any other single factor, has begun to influence not only the range of jobs to which data management is being applied, but the architecture of the systems themselves. This is not surprising. In any field, increasing demand for a product or service has a strong influence on the basic economics of providing that product or service, and a subsequent change in the methods of constructing and supplying it. Probably the most obvious example of this in data processing is the history of operating systems. Early operating systems were little more than simple utilities which functioned as aids in using the hardware. Today they are a necessity. Ten years ago it was extremely difficult to imagine a manufacturer supplying as a standard operating system something as extensive and complex as the System/360 Operating System. Now the capabilities of such systems are regarded as commonplace, and users have developed even higher expectations for the future.

The motivation for the development of these vastly improved operating systems has rested largely on the growth of the data processing industry. This has allowed manufacturers to make major investments to develop such complex systems and to amortize their cost over a much wider market and a long period of time.

Data management systems have now begun a transition very similar to that of operating systems. They are still largely in the category of a utility, albeit a rather sophisticated one. The volume of demand for their capabilities, however, is sufficiently high so that new DMS architectures for meeting this demand efficiently can now be realistically considered.

The central thesis of this paper is that the most effective way of supplying data management capabilities in the future is to imbed a substantial number of the components of a DMS within a modern operating system. The specific components chosen in this paper are those which, in addition to being common to every DMS, are sufficiently alike in current data management systems to warrant use of a single implementation of that component in future systems. It is argued that this approach provides the following benefits:

i. Any component of a data management system can be made to operate more efficiently and effectively when it is included as part of the operating system.

ii. Many DMS components and the functions they support are not unique to data management systems but are common to many other types of processing. Consequently, inclusion of DMS components within the operating system makes them more widely available, and promotes compatibility between the data management system and other non-DMS programs.

iii. Given adequate implementation of these components, the designers of data management systems will be free to concentrate on providing better capabilities in support of specific DMS applications.

A straightforward way of defending these statements is to describe the problems faced by designers of early data management systems in terms of the tools available to them, and to compare these problems to those faced by current designers and implementors of data management systems. This comparison reveals a trend toward providing many facilities which are essential in the implementation of data management systems within current operating systems. By extrapolation of this trend, a probable architecture for future data management systems can be described. This description substantiates the main thesis of this paper.

DMS IN THE PAST

The most striking problem facing designers and implementors of data management systems in the early 1960's was the lack of almost any basic support software useful for building a data management system.

The author of this paper was a member of a group which implemented a data management system on the SAGE computer, the AN-FSQ-7, in 1961. This machine had extremely impressive hardware capabilities for its time, and provided an excellent vehicle for constructing a data management system. It had 65k, 32 bit words of 6μs core storage, high speed drum storage with an access time of under twenty milliseconds, a communications drum which allowed for simultaneous use up to 10 teletypes, graphics display consoles, and high speed electrostatic printers. This system, however, was devoid of any general purpose software. There was no operating system or utility package available, other than a symbolic assembler and loader.

The data management system which was built, called ETF, was an experimental system and was not intended for operational use, although it was used as a base for prototype applications in military airlift operations and in post-attack command and control. The data management capabilities which this system provided included a general purpose query language and output formatting routines, file structures providing rapid retrieval through a hash-code technique and multi-terminal on-line operation. In many respects, this system was comparable in capabilities and performance to some that are being marketed today. The process of implementing it, however, was considerably different.

This implementation process can be conveniently divided into three phases. First was the implementation of very low-level routines such as input-output handling and core storage allocation. These can be thought of as "basic operating system components". Today they are taken completely for granted. Second were file and dictionary access programs, file storage allocators for management of disk space, and routines to manage multiple simultaneous teletype input-output including management of task queues. These routines can be termed "basic DMS components". At the time, most of these capabilities were quite new and were largely untried. Today, they are taken for granted in almost every DMS, although they are built using a variety of implementation techniques.

Once the capabilities described above had been provided, the third phase could begin. This phase concentrated on the implementation of the query language translator, output formatting routines, special computational routines and the file generation program. These programs can be termed "basic language components". This division of a DMS into components can be further clarified by thinking of them in terms of those which are transparent to a user of the DMS and those which are visible to him. The operating system components and basic DMS components are transparent to a user of the system, although they have a strong influence on its performance. The basic language components constitute the user's interface with the data management system and strongly influence the suitability of the system for various applications.

A further observation about the ETF system is that the basic language components of the system were by comparison easier to construct than were the basic DMS components. This is true because the presence of an adequate set of basic components allows for substantial change within the language components of the system even after their initial implementation. The most important observation, however, is that virtually the entire system had to be built from scratch.

CURRENT DATA MANAGEMENT SITUATION

All of the goals which the designers of the ETF system were trying to achieve are still very valid. Such capabilities as on-line query languages supported by high-speed retrieval mechanisms are characteristics which many systems today either contain or aspire to. The MRI Inc. System 2000, Cambridge Computer Associates CCA104 System and TRW's GIM System are all contemporary systems which were designed with these characteristics as major goals. Systems such as IBM's NIPS and Informatics MARK IV which were fairly simple batch processing

systems in their original implementations have evolved upwards in complexity to include many of the features of newer higher performance systems.

In addition to similar goals, the design techniques used in current systems are remarkably like those employed in early systems. This is not to say that there have not been improvements, because there have, however, such file access methods as hash-coding, inversion of files using bit vector schemes or pointers had all been implemented at least once by 1962. Current query languages also show remarkable similarity to those of early systems. For instance, the language of SDC's DS/2 system bears more than a vague resemblance to that of the ETF system.

Where have the major advances in data management systems ocurred, then? There have been no dramatic breakthroughs in the field of technical design. Improvements have been evolutionary in nature and have come largely through reimplementation and refinement of basic techniques. The most significant advance in data management systems is their acceptance by a much broader segment of the data processing industry than was formerly the case, and the dramatic rise in the number of available systems. Much of the reason for this availability is due to the inclusion of basic tools within the operating system, similar to the basic DMS components described above, which makes implementation of data management systems quicker and cheaper.

There are a number of examples of basic DMS components which are contained in current operating systems. The most important of these is physical file access methods. It is important because the designer of any data management system must decide whether to make use of access methods which have been provided with the operating system or to design and implement his own. The cost differential between these alternatives is large, as is the implementation time. Because of this, many data management systems make use of these access methods. Within the IBM 360/370 environment, the NIPS system utilizes the Indexed Sequential Access Method for organizing its data files. This is also true for the Systems Development Corporation's DS/2 system which offers the option of using either sequential or indexed sequential access methods. The INQUIRE data management system built by Infodata Systems, Inc., for use on IBM 360/370 equipment, uses a combination of indexed sequential and direct access methods. The data management system bid by Honeywell Information Systems in its WWMCCS submission provides an additional example. Originally, this system was designed to handle tape files compatible with a number of other systems including COBOL. As a part of its enhancement for the WWMMCS bid, interfaces between the system and additional access methods available under GECOS III were built. These include both

the random file capability as well as the recently announced Honeywell Indexed Sequential Access Method.

There are numerous other examples of the type cited above, all of which serve to demonstrate that current data management system designers frequently make extensive use of manufacturer supplied access methods.

A second significant area in which operating system capabilities are used to support data management functions is in teleprocessing. The use of IBM's teleprocessing access methods by many existing data management systems is an example of this. While these capabilities are rather basic, the availability of the Time Sharing Option of OS 360 provides a much more substantial set of such tools and will certainly be used in future systems. This can be illustrated most clearly with an example drawn from the author's recent experience.

The Air Force Data Services Center (AF/ACS) established a project in 1970 to acquire an interim data management system. The system which was chosen had been built by General Electric Apollo Systems Department for NASA and was called ADVISOR. The system had been implemented originally under GECOS II, which had no time-sharing subsystem. The system, however, was intended to support multiple on-line terminals. The implementors were naturally forced to develop a sub-monitor of their own since GECOS II provided no support for subsystems with terminal capabilities. The GECOS II version of ADVISOR, including its sub-monitor, required 34K words of core storage. Furthermore, it was necessary to alter the job scheduling algorithms of the GECOS II operating system to guarantee adequate response times to ADVISOR terminals. At the time the ADVISOR system was selected for installation at AF/ACS, GECOS III with its time-sharing subsystem had become available, and it was decided to modify the ADVISOR system to run under time-sharing.

The modifications to the system resulted in substantial improvement in two areas. First, the amount of core required by the system went from 34K words to 23K words, a reduction of almost one-third. Second, the response time at terminals was significantly better in the modified version even when the DMS was competing with a number of other time-sharing users.

It is the author's belief that this experience is representative of the kinds of savings which could be realized in many data management systems by using time-sharing facilities supplied as a part of the operating system. This does not mean that current time-sharing systems provide all features needed for support of data management systems. However, they represent a significant advance over tools which have been available in the past.

CURRENT TREND

The examples presented above have illustrated the trend towards including many basic DMS components necessary to data management within operating systems. Clearly, some of these components would exist in the absence of any DMS requirements. However, a large enough number of them have been provided specifically for data management related functions so that a precedent has been established. This raises two additional questions. First, will this trend continue? And, second, should it continue?

Will Trend Continue?

The answer to the first question rests on two main issues: the additional needs of data management system designers in terms of basic DMS components required, and the size of the market perceived for these by suppliers of operating systems. The usefulness of additional components for use in constructing data management systems can be shown by an examination of a number of specific representative areas in which such components are now being developed. These are discussed in more detail in the next section. The accurate determination of the size of future markets for these components is more difficult. The president of IBM, Mr. Cary, in a recent article[1] is quoted as saying that systems incorporating a data base and data communications have the most future market potential. A major manufacturer of data processing equipment is known to be developing a large array of tools for data base manipulation which are architecturally within the operating system. At a general level, the increase in the amount of formatted file processing and more specifically the processing of large shared data bases, is an established fact. The wide use of systems such as IBM's IMS and the CINCOM's TOTAL system as a set of basic DMS components for data base management systems bears this out.

Should Trend Continue?

The second question raised above; namely, the desirability of a continuation of this trend is largely a subjective question. In discussing data management system design with collegues, this author has more than once heard the argument advanced that "operating systems are making it difficult to implement many of the low-level detailed functions required by data management". This is undoubtedly true, and is in fact additional evidence that the trend cited in this paper is established. These people, to a great extent, are concerned with the specific problem of having to deal with standard access methods which do tend, in some instances, to make it difficult to develop complex new file structures. The alternative to this is to

provide features within the operating system which will allow implementors to start from scratch at a "bare bones" level and build their own systems. This is incompatible with the fundamental philosophy of modern operating systems. Multiprogramming systems, for example, require that users give up some individual freedoms (e.g., first-level interrupt handling) in the interests of overall system efficiency. In the case of file access methods, their use on a wide basis indicates the existence of a concensus that the availability of a standard set of them is sufficiently valuable to warrant a compromise in flexibility. None of these statements is intended to suggest that data management system designers should be prevented from experimenting with new techniques. This, in fact, should be encouraged. It does mean, however, that much of this experimentation may have to be carried out external to the environment of standard operating systems. In short, the trend toward the wider availability of basic DMS components provides useful tools to implementors of operational data management systems on standard hardware and software and can sometimes be a hinderance to those designers whose interests lies solely in experimentation with new techniques on these same systems.

To return to the main question at hand, the established trend toward placing basic DMS components in operating systems should continue because it shows every indication of further lowering the implementation cost of data management systems, allowing the implementors of the DMS to concentrate on basic language components, and increasing the compatibility among all software elements of any facility using the standard operating system. The following section describes a number of specific technical areas on which a continuation of the trend is likely to be based.

SOFTWARE CAPABILITIES FOR INCLUSION IN FUTURE OPERATING SYSTEMS

Each of the topics discussed below is an area in which there is currently one or more implementation activities underway or which are being widely and actively studied. These areas can be divided into two main types: those which were originally motivated by or developed specifically for data management use, and those which were or are being developed for general use but are critically important to future data management systems. In either case, they represent basic DMS components which belong within an operating system.

Improved File Structures

The need for increasingly sophisticated file structures is well established. The inclusion of support for these structures within

the operating system is advantageous for two main reasons. First, it makes such structures available to all processing functions, including procedure oriented languages. Second, such structures can be provided through extension of current operating system capabilities. These extensions are described in detail below.

Before discussing specific file structures, definitions of logical and physical file structures are appropriate, since there is often confusion over the distinction between these terms. For the purposes of this paper, a logical file structure is the set of expressed or implied relationships between records or entities within a file or data base. Thus, such terms as "flat files", "hierarchical files", and "network structured files" all describe logical file structures. Physical file structures are the mechanisms utilized to store and retrieve file data. These include such techniques as sequential access methods, direct access methods, bit vector schemes for file inversion, and others. It should be noted that there is usually, but not necessarily, a one to one correspondence between physical and logical file structures. For example, an indexed sequential access method might be used to store either flat or hierarchical files.

Symbolic Reference to Data

A conceptually simple extension of current operating systems' file access methods is that of symbolic field-level reference to data. All current access methods are organized at the lowest level around the notion of records. In only limited cases (the index of ISAM is the most obvious example) is the access method sensitive to units of information at a level lower than that of a record. Both data management systems and formatted file related applications programs operate primarily on field level information which in most cases is represented as contiguous strings of characters within a record. Furthermore, most non-DMS applications programs are bound to specific record formats in the sense that any change in either field level information or the physical file structure causes the program to run incorrectly. In the past, this situation was undesirable but could be tolerated, since many application programs were the sole users of a data file. In a shared data environment, this method of binding programs to specific formats is highly undesirable. Data formats are changed quite frequently, forcing application programs to change with them. What is needed is an ability to maintain a single description of a record's contents rather than allowing each applications programmer to maintain a separate and unique data declaration. This situation has been available in COBOL by using external data divisions, however, it is rarely taken advantage of.

A more desirable mechanism for both data management systems and non-DMS applications programs is a physical data description or data declaration managed by the operating system as a natural extension of current access methods. This is very similar in concept to the dictionary of current data management systems, and its inclusion as a part of file manipulation routines makes its advantages more widely available. An applications programmer or data management system making use of this capability would invoke the name or identifier of the record type it desired access to, and after reading any record of this type, the fields of that record could be referenced symbolically. The most obvious way of providing this capability in the future is to extend, to the field level, the file cataloging and file directory services provided by current third-generation operating systems.

Physical File Structures

The key to the performance of any data management system is the physical file structure it employs. Because of this, designers of data management systems have devoted more time and thought to the relationship between logical and physical file structures than any other single component in a DMS. There are a number of generally accepted maxims which have evolved from the observation of many implementations of different types of file structures. The most important by far is that there is no single access technique or physical file structure which provides optimum performance over a wide range of common file processing applications. For example, demands for very high speed retrieval from large files are incompatible with those of large volume, rapid updating, and high volume output requests. To a great extent, this is a reflection of the limitations of current secondary storage devices, and it points out the needs for a diversity of access methods from which the most suitable one for any application can be selected.

Because no single physical access method stands out as being universally better than all others, it is difficult to propose a specific one, or even a small set for inclusion in future operating systems. It is clear, however, that demand for increasingly complex physical structures is increasing, indexed sequential and direct access being the best examples, and is likely to continue. Perhaps the most obvious next steps in complexity are those of multiple indexes for a single file (file inversion) and the provision of physical structures which support logical structures of increased complexity.

File inversion techniques, and there are many of them, are the most frequently used mechanism for providing high-speed retrieval within the limitations of currently available hardware. For applications requiring this type of performance, a rapid method of selecting records based on their content is necessary. To either a data

management system designer or to an applications programmer, the capability would take the general form of a call to the operating system which delivers to the user the "next" record or records whose contents meet the criteria specified in the call. The specific physical access methods used within the operating system to achieve this should be largely transparent to the program making the call. Furthermore, it should be mentioned that the usefulness of such a capability is made possible by the availability of the file dictionary described previously, since any qualifying field can be referenced symbolically.

Logical File Structures

In addition to improved physical access methods, an ability to manipulate file structures of greater logical complexity is needed. This need is currently being met by such systems as IMS and TOTAL, which support hierarchical and network structured files. An example of the need for these structures can be found in the Military Airlift Command's MACIMS system. One of the primary goals of this system is the management of a large body of interrelated data. This includes a reduction in the redundancy of data storage, and the ability to provide data to a number of different functional processors, each requiring a different but, not necessarily unique, subset of the data. Comparison of these requirements with the abilities of traditional file structures such as that supported by COBOL reveals a significant gap between requirements and capabilities, and thus a need for structures such as those provided by IMS.

The implementation of support for such structures within an operating system is reasonably straightforward technically, and can be constructed using the dictionary and some of the physical access capabilities proposed earlier. The most difficult problem currently being faced is that of arriving at a consensus as to which set of logical and physical structures are best. It seems likely that much of the discussion is due to a lack of sufficient data on the relative merits of one structure versus another, and that any firm decision cannot be made until such data is generated.

Regardless of what structures are chosen, the effects on the operating system and the types of support it must provide are much the same. At a detailed level, the capability needed is that of retrieving a record from a file based on a logical relationship of of that record to another in the file. Thus, a simple example is one of moving downward in a hierarchical tree to retrieve the offspring records of a parent record. Again an important point about placing these capabilities within the operating system is that they become available not only to data management system designers and implementors

as a set of capabilities which form a base on which a data management system can be constructed, but also to the application programmer as a basic extension of his programming language.

Additional Basic DMS Components

The discussion above on file structures is largely centered on techniques which have been developed within the domain of data management, but which today find much broader applicability. The additional areas discussed below originated in a wider context than DMS but are critical to their operation and which most sensibly (from a technical and economic point of view) belong in the domain of the modern operating system.

Concurrent Task Management

Many data management systems are designed with at least some thought of providing service to multiple simultaneous on-line users. As the examples in the earlier portions of this paper have shown, one promising vehicle for providing this capability is the time-sharing subsystem of many modern operating systems.

Unfortunately, current time-sharing systems are not entirely adequate for supporting all of the concurrent task management needs of a data management system. In particular, most time-sharing systems have been developed to support independent, unrelated jobs. This is incompatible with many needs of current data management systems. A prime example is the area of related job scheduling. In many data management applications, it is desirable to be able to assign priority to different types of users or the jobs which they are performing. This need is usually met by providing a method of queueing a set of jobs all operating on a single file, and interrupting tasks which are in execution to service, on arrival, higher priority tasks. Current time-sharing systems must be improved substantially in this area to meet the needs of data management systems. An additional improvement to time-sharing systems which would benefit not only data management systems but all users of time-sharing is that of reentrant program support. In a data management system environment, there is a high probability that more than one user is executing the same module of the data management system. For example, in a system with ten simultaneous on-line users, it is very likely that more than one of them is making use of the query language translator. This is a situation where reentrant programs can provide a significant saving in core space required. In the operation of some current data management systems on IBM 360 equipment, it is always distressing to be forced to maintain a separate copy of the data management system in core for each system user. The effective use of reentrant programs running in a time-sharing environment can eliminate much of this problem.

Current time-sharing systems, then, must be improved to allow data management systems to operate at a high level of efficiency, and with a full set of capabilities. Nevertheless, the use of current time-sharing systems offers considerable advantage over the alternative of building similar capabilities nearly from scratch.

Management of Concurrent File Access

The problem of managing concurrent file access, including both reading and writing of files, has no general solution. This is a problem which is of interest to not only data management systems, but any set of programs referencing a common set of data files. There are a number of examples which can be used to illustrate the problem, probably the simplest of which is the case where program A is updating a file, and program B is reading it. Both programs are operating in a multiprogramming environment where either may be interrupted by the operating system without notice. Program A, then, may be stopped in the midst of an update which changes not only a data value, but the structural integrity of the file. Program B, in trying to read the file, may find a logically inconsistent data structure which it cannot sensibly process.

The general solution to this problem at the level of small units of file access, such as records, is not known. However, it is clear that any solution will be an integral part of the operating system. It must be tightly bound to the file access mechanisms of the system in order to maintain an awareness of all accesses to a given file. One can imagine a central, perhaps reentrant, routine which maintains control tables for all file access by any program or system. It is in a position to detect conflicts, and schedule file accesses in such a way that the conflict is eliminated. The need for such a solution becomes more critical as shared data applications come into wider use. Current solutions to the problem such as preventing all file access from being made while an update process runs to completion simply are not satisfactory.

Secure Data Management System Operation

A problem which is of obvious special interest to the DOD community is that of providing secure operation in a multi-access computing facility. This problem is a general one, and affects not only data management systems, but all programs running in a facility handling classified information. The problem is of particular interest to designers, implementors, and users of data management systems, since the central objective of a secure facility is protection of file data from unauthorized or accidental disclosure. Like all other functions discussed in this section, a majority of the mechanisms for providing

secure operation must be within the operating system. The reason for this is straightforward. Assume the existence of a data management system with an arbitrary number of security controls built into it. Also assume that this system runs under the control of an operating system which has not been designed to be secure. In such a situation, it is a simple matter for any person with a system programmer's knowledge to access data belonging to the data management system through independent means. Consequently, a secure data management system is one which maintains its own security controls, but depends primarily on its operating system for secure operation.

EFFECTS OF PLACING BASIC DATA MANAGEMENT SYSTEM COMPONENTS WITHIN THE OPERATING SYSTEM

The value to data management system designers and implementors of the capabilities discussed above can be shown by considering again the process of constructing the capabilities provided in the early ETF system, and comparing this with the original example. The most dramatic change is the virtual absence of both the first and second phases of the original implementation. The first phase is made unnecessary because all of the functions originally implemented in it are available today in modern operating systems. The second phase has been largely eliminated because the basic DMS components would be available within the operating system. This phase now becomes largely a process of selecting from available components, those most appropriate to the characteristics of the system being constructed. In the case of ETF, the first step would be the selection of a prime access method. Any key value or index method providing rapid access to individual records by name would likely duplicate (or better) the performance of the original system. The logical structure required would be a two-level hierarchical file, which compared to some structures supported even today, such as in IMS, is nearly trivial, and would certainly be available within the standard structures supported in the future. Having chosen one (or possibly more than one) access method and determined that the file structures desired were supported, design could begin on the query language translator, output formatting routines, retrieval processors, and file generation routines. The development of these programs could be done using the same time-sharing system that would ultimately support the data management routines as one of its subsystems. This is particularly advantageous since time-sharing is an excellent tool for program development and can reduce the elapsed time involved by 50 per cent or more.[2,3]

After completing these functions and installing those that interfaced with on-line users under the time-sharing system, the new ETF system would be complete. In the author's opinion, this procedure would provide a savings in time and manpower of roughly one-half over that of the original system.

In addition to the savings in implementation costs, the system of the previous example and for that matter any data management system built from the same basic capabilities would have a number of significant additional advantages over current and previous systems. First is the relationship between the data management system and standard programming languages. Since both the DMS and standard compilers would use the dictionary capability described earlier, data files operated on by the data management system could also be directly accessed from any programming language within the system. In the past, this sort of compatibility at the data level has often been desirable but largely unavailable. No problem-oriented language such as those provided in many present data management systems is capable of supporting very complex processing. Occasionally, this type of processing is necessary. However, in any data management system with a unique file structure, processing of its data with external procedure oriented language programs becomes nearly impossible. The common use of the data dictionary eliminates much of this problem.

The second advantage of data management systems built with basic capabilities within the operating system, is that of better overall stability. Any capability included as part of a standard operating system receives better maintenance, is better documented, and has better training associated with it than it would as a user generated collection of software. This is largely due to economy of scale; widely used software generates increased revenues, however, maintenance and documentation costs are largely independent of the number of users of the software.

CONCLUSIONS

Data management today is experiencing more rapid acceptance of its capabilities than has ever been the case in the past. The experience gained within the DOD community over the past ten years in DMS has been responsible to a great extent for this situation. This acceptance provides the possibility of altering the basic architecture of data management systems so that many of its components can be included within the operating system, thereby providing better support and wider availability for them. The inclusion of many basic DMS capabilities in the operating system can only be done successfully, however, by selecting broadly useful techniques which have been proven out in a number of implementations in the past.

The overall goal of the DMS community should be to make data management a naturally available tool within any data processing facility rather than a separate, special purpose collection of software as it has so often been in the past. Its potential value to a broad range of data processing problems is now established and accepted. The realization of this value is the most important problem facing data management.

REFERENCES

1. A. Pantages, R. B. Frost, "IBM: Changes at the Top" Datamation Vol. 17, No. 21, November 1, 1971, pp. 26-28.

2. M. Schatzoff, R. Tsao, R. Wiig, "An Experimental Comparison of Time-Sharing and Batch Processing", Communications of the ACM, May 1967.

3. H. Sackman, W. J. Erikson, E. E. Grant, "Exploratory Experimental Studies Comparing Online and Offline Programming Performance", Communications of the ACM, January 1968.

BIBLIOGRAPHY

System Development Corporation, DS/2 Users Manual, Santa Monica, California, 1970.

Computer Corporation of America, Technical Reference Manual, Series 100, Information Retrieval Software Systems, Models: 102, 104, Cambridge, Massachusetts, October 1, 1970.

TRW Systems Group, GIM System Summary, TRW Document No. 3181-A, Revision 01, Redondo Beach, California, 15 August 1969.

IBM Federal Systems Division, System/360 Formatted File System (NIPS), Vol. I-VIII, 30 September 1969.

National Aeronautics and Space Administration, MA001-013-1, ADVISOR MSF-DPS Familiarization Manual, Washington, D. C., January 1970.

IBM Corporation, GH20-0765-1, Information Management System/360, Version 2 General Information Manual, 1971.

Cincom Systems, Incorporated, TOTAL - the Data Base Management System, Reference Manual, Edition II, Version I, 1971.

Informatics Inc., Document No. SP-70-810-200, MARK IV File Management System, 1970.

MRI Systems Corpration, Document No. RM S2K 2.1, System 2000 Reference Manual, 15 July 1971.

Infodata Systems Inc., Inquire Technical Summary, April 1970.

IBM Corporation, Document No. GC26-3746-0, IBM System/360 Operating System Data Management Services, January 1971.

CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems, Association for Computing Machinery, New York, New York, April 1971.

CODASYL, Data Base Task Group Report to the CODASYL Programming Language Committee, Association for Computing Machinery, New York, New York, April 1971.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
The MITRE Corporation P.O. Box 208 Bedford, Mass.	UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

CURRENT TRENDS IN DATA MANAGEMENT SYSTEM ARCHITECTURE

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

J.A. Singer

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
MARCH 1973	22	3

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F19(628)-71-C-0002 b. PROJECT NO. 572M c. d.	ESD-TR-72-157 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) MTR-2303

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Deputy for Command and Management Systems Electronic Systems Division, AFSC L.G. Hanscom Field, Bedford, Mass.

13. ABSTRACT

Early data management system architecture is examined and compared with that of current systems. This comparison reveals a trend toward placing a number of basic data management system components within the modern operating system. A continuation of this trend is postulated and a number of specific examples of areas where the trend is likely to continue are given. The advantages of a continuation of this trend are described in terms of both the ease of building new data management systems and the increased compatibility of such systems with other data processing tools.

DD FORM 1 NOV 68 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
DATA MANAGEMENT CAPABILITIES						
DATA MANAGEMENT SYSTEMS						
ETF SYSTEM						
OPERATING SYSTEMS						